```
@ Filename: agarwal4.s
@ Author:   Vasu Agarwal - vra0004@uah.edu - CS 413-01 spring 2023
@ Purpose:  The objective of this assignment is to simulate the operation of a
vending machine.
@     The machine will dispense, upon reception of the correct amount of money, a
choice of Gum,
@     Peanuts, Cheese Crackers, or M&Ms. Also displaying output using the LED's

@-------------------------------------------------------------------------------
---
@ Use these commands to assemble, link, run and debug this program:
@   First run the following to get superuser access.
@   "sudo su" is the command to allow running without having to
@    use sudo.
@    as -o agarwal4.o agarwal4.s
@    gcc -o agarwal4 agarwal4.o -lwiringPi
@    ./agarwal4 ;echo $?
@    gdb --args ./agarwal4
@-------------------------------------------------------------------------------
----

@SECRET CODE: S


OUTPUT = 1 @ Used to set the selected GPIO pins to output only.
.equ READERROR, 0
.text
.global main

@@@@@
main:
@@@@@

@@@@@@@@@
setAmount:
@@@@@@@@@
      mov r4, #2                @ Gum count
      mov r5, #2                  @ Peanuts count
      mov r6, #2                  @ Crackers count
      mov r7, #2                  @ M&M count

@@@@@@
setup:
@@@@@@


      bl      wiringPiSetup

        mov     r1,#-1

        cmp     r0, r1

        bne     init  @ Everything is OK so continue with code.

        ldr     r0, =ErrMsg

        bl      printf

        b       errorout  @ There is a problem with the GPIO exit code.
```

```
@@@@
init:
@@@@
        ldr     r0, =pin2

        ldr     r0, [r0]

        mov     r1, #OUTPUT

        bl      pinMode

@ set the pin3 mode to output


        ldr     r0, =pin3

        ldr     r0, [r0]

        mov     r1, #OUTPUT

        bl      pinMode

@ set the pin4 mode to output


        ldr     r0, =pin4
        ldr     r0, [r0]

        mov     r1, #OUTPUT

        bl      pinMode

@ set the pin5 mode to output


        ldr     r0, =pin5

        ldr     r0, [r0]

        mov     r1, #OUTPUT

        bl      pinMode

@ Write a logic one to turn pin2 to on wait 5s then to off.


        ldr     r0, =pin5
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite

      ldr     r0, =delay5Ms
        ldr     r0, [r0]
        bl      delay

            ldr     r0, =pin5
        ldr     r0, [r0]
        mov     r1, #0
```

```
        bl      digitalWrite


@@@@@@
prompt:
@@@@@@
      ldr r0, =strInputPrompt     @ Welcome message with instructions attached
      bl printf

      ldr r0, =userInput           @ Read user's input for selection
      ldr r1, =numInput
      bl scanf
      cmp r0, #READERROR
      beq readerror
      ldr r1, =numInput
      ldr r1, [r1]

      @ Section checks what user selected, then branches correctly
      cmp r1, #'G'
      beq gum
      cmp r1, #'P'
      beq peanuts
      cmp r1, #'C'
      beq crackers
      cmp r1, #'M'
      beq mnms
      cmp r1, #'S'
      beq printInventory          @ Secret Code is checked, branching to
inventory('S')
      b prompt

@@@@
gum:
@@@@
      ldr r0, =userSelection
      ldr r1, =itemGum
      bl printf                   @ Confirmation Check
      ldr r0, =userInput
      ldr r1, =numInput
      bl scanf                    @ Reads y or n from user
      cmp r0, #READERROR
      beq readerror
      ldr r1, =numInput
      ldr r1, [r1]

      cmp r1, #'y'
      bne prompt                  @ If not y return to prompt
      cmp r4, #0
      beq emptyInventory               @ Check and branch if no inventory left
      sub r4, r4, #1              @ Else, we subtract from inventory

@@@@@@@@@@@@@
inventoryGum:
@@@@@@@@@@@@@
      mov r1, #50
      push {r1}                   @ Pushes the cost of gum (50 cents)
      b popper

@@@@@@@@
```

```
peanuts:
@@@@@@@@

      ldr r0, =userSelection
      ldr r1, =itemPeanuts
      bl printf
      ldr r0, =userInput
      ldr r1, =numInput
      bl scanf
      cmp r0, #READERROR
      beq readerror
      ldr r1, =numInput
      ldr r1, [r1]

      cmp r1, #'y'
      bne prompt
      cmp r5, #0
      beq emptyInventory
      sub r5, r5, #1

@@@@@@@@@@@@@@@@
inventoryPeanuts:
@@@@@@@@@@@@@@@@

      mov r1, #55
      push {r1}
      b popper

@@@@@@@@
crackers:
@@@@@@@@
      ldr r0, =userSelection
      ldr r1, =itemCrackers
      bl printf
      ldr r0, =userInput
      ldr r1, =numInput
      bl scanf
      cmp r0, #READERROR
      beq readerror
      ldr r1, =numInput
      ldr r1, [r1]

      cmp r1, #'y'
      bne prompt
      cmp r6, #0
      beq emptyInventory
      sub r6, r6, #1

@@@@@@@@@@@@@@@@@
inventoryCrackers:
@@@@@@@@@@@@@@@@@

      mov r1, #65
      push {r1}
      b popper

@@@@@
mnms:
@@@@@
```

```
      ldr r0, =userSelection
      ldr r1, =itemMNMs
      bl printf
      ldr r0, =userInput
      ldr r1, =numInput
      bl scanf
      cmp r0, #READERROR
      beq readerror
      ldr r1, =numInput
      ldr r1, [r1]

      cmp r1, #'y'
      bne prompt
      cmp r7, #0
      beq emptyInventory
      sub r7, r7, #1

@@@@@@@@@@@@@
inventoryMNMs:
@@@@@@@@@@@@@
      mov r1, #100
      push {r1}
      b popper

@@@@@@@@@@@@@@@
emptyInventory:
@@@@@@@@@@@@@@@


      ldr r0, =noInventory
      bl printf                   @ Prints that there is no inventory left for an
item
      b prompt

@@@@@@@
popper:
@@@@@@@
      pop {r8}                    @ Pops the original price of an item into r8
      mov r11, r8                 @ Puts a copy in r11

@@@@@@@@@
inventory:
@@@@@@@@@
      ldr r0, =userPayment
      mov r1, r8
      bl printf                   @ Prompts the user to enter x cents

      ldr r0, =userInput
      ldr r1, =numInput
      bl scanf                    @ Reads in (D, Q, B) as the change entered
      cmp r0, #READERROR
      beq readerror
      ldr r1, =numInput
      ldr r1, [r1]

      cmp r1, #'D'
      beq dime
      cmp r1, #'Q'
      beq quarter
```

```
      cmp r1, #'B'
      beq dollarbill

@@@@@
dime:
@@@@@
      sub r8, r8, #10             @ Subtracts 10 cents if a dime is entered
      cmp r8, #0                  @ If the total cost remaining has reached zero,
end loop
      ble change
      b inventory                 @ Else, continue loop

@@@@@@@
quarter:
@@@@@@@
      sub r8, r8, #25
      cmp r8, #0
      ble change
      b inventory

@@@@@@@@@@@
dollarbill:
@@@@@@@@@@@
      sub r8, r8, #100
      cmp r8, #0
      ble change
      b inventory

@@@@@@
change:
@@@@@@
      ldr r0, =enoughPayment
      bl printf                   @ Informs user that enough payment has been
provided
      cmp r11, #50
      beq printgum
      cmp r11, #55
      beq printpeanuts
      cmp r11, #65
      beq printcrackers
      cmp r11, #100
      beq printmnms

@@@@@@@@
printgum:
@@@@@@@@

      ldr     r0, =pin5
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

            ldr     r0, =pin5
        ldr     r0, [r0]
        mov     r1, #0
```

```
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

      ldr     r0, =pin5
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

            ldr     r0, =pin5
        ldr     r0, [r0]
        mov     r1, #0
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

      ldr     r0, =pin5
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite

      ldr     r0, =delay5Ms
        ldr     r0, [r0]
        bl      delay

            ldr     r0, =pin5
        ldr     r0, [r0]
        mov     r1, #0
        bl      digitalWrite

      ldr r0, =dispensed
      ldr r1, =itemGum
      bl printf                   @ Prints that item has been successfully
dispensed
      b changeoutput

@@@@@@@@@@@@
printpeanuts:
@@@@@@@@@@@@

      ldr     r0, =pin4
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

            ldr     r0, =pin4
        ldr     r0, [r0]
```

```
        mov     r1, #0
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

      ldr     r0, =pin4
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

          ldr     r0, =pin4
        ldr     r0, [r0]
        mov     r1, #0
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

      ldr     r0, =pin4
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite

      ldr     r0, =delay5Ms
        ldr     r0, [r0]
        bl      delay

          ldr     r0, =pin4
        ldr     r0, [r0]
        mov     r1, #0
        bl      digitalWrite

      ldr r0, =dispensed
      ldr r1, =itemPeanuts
      bl printf
      b changeoutput

@@@@@@@@@@@@@
printcrackers:
@@@@@@@@@@@@@
      ldr     r0, =pin3
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

          ldr     r0, =pin3
        ldr     r0, [r0]
        mov     r1, #0
```

```
  bl      digitalWrite

ldr     r0, =delayMs
  ldr     r0, [r0]
  bl      delay

ldr     r0, =pin3
  ldr     r0, [r0]
  mov     r1, #1
  bl      digitalWrite

ldr     r0, =delayMs
  ldr     r0, [r0]
  bl      delay

      ldr     r0, =pin3
  ldr     r0, [r0]
  mov     r1, #0
  bl      digitalWrite

ldr     r0, =delayMs
  ldr     r0, [r0]
  bl      delay

ldr     r0, =pin3
  ldr     r0, [r0]
  mov     r1, #1
  bl      digitalWrite

ldr     r0, =delay5Ms
  ldr     r0, [r0]
  bl      delay

      ldr     r0, =pin3
  ldr     r0, [r0]
  mov     r1, #0
  bl      digitalWrite

ldr r0, =dispensed
ldr r1, =itemCrackers
bl printf
b changeoutput

@@@@@@@@@
printmnms:
@@@@@@@@@

ldr     r0, =pin2
  ldr     r0, [r0]
  mov     r1, #1
  bl      digitalWrite

ldr     r0, =delayMs
  ldr     r0, [r0]
  bl      delay

      ldr     r0, =pin2
  ldr     r0, [r0]
  mov     r1, #0
```

```
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

      ldr     r0, =pin2
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

          ldr     r0, =pin2
        ldr     r0, [r0]
        mov     r1, #0
        bl      digitalWrite

      ldr     r0, =delayMs
        ldr     r0, [r0]
        bl      delay

      ldr     r0, =pin2
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite

      ldr     r0, =delay5Ms
        ldr     r0, [r0]
        bl      delay

          ldr     r0, =pin2
        ldr     r0, [r0]
        mov     r1, #0
        bl      digitalWrite

      ldr r0, =dispensed
      ldr r1, =itemMNMs
      bl printf
      b changeoutput

@@@@@@@@@@@@
changeoutput:
@@@@@@@@@@@@
      ldr r0, =changeOutput
      mov r1, r8
      mov r9, #-1                 @ Makes negaive number positive to represent
change
      mul r10, r8, r9
      mov r1, r10
      bl printf                   @ Prints the amount of change returned

@@@@@@@@@@@@@@@
checkinventory:                 @ Checks the inventory of each item to check if
program should continue
@@@@@@@@@@@@@@@
      cmp r4, #0
```

```
      bne prompt
      cmp r5, #0
      bne prompt
      cmp r6, #0
      bne prompt
      cmp r7, #0
      bne prompt
      ldr r0, =noInventory
      bl printf
      b myexit

@@@@@@@@@@@@@@
printInventory:                 @ Section for secret input
@@@@@@@@@@@@@@
      ldr r0, =secretInventory
      mov r1, r4
      mov r2, r5
      mov r3, r6
      bl printf
      ldr r0, =mnmInventory
      mov r1, r7
      bl printf
      b prompt

      b myexit

@@@@@@@@@
readerror:
@@@@@@@@@
      ldr r0, =strInputPattern
      ldr r1, =strInputError
      bl scanf
      b prompt

@@@@@@
myexit:
@@@@@@

      ldr     r0, =pin5
        ldr     r0, [r0]
        mov     r1, #1
        bl      digitalWrite
      ldr     r0, =delay5Ms
        ldr     r0, [r0]
        bl      delay
            ldr     r0, =pin5
        ldr     r0, [r0]
        mov     r1, #0
        bl      digitalWrite

      mov r7, #0x01
      svc 0


.data

.balign 4
strInputPrompt: .asciz "\nWelcome to the vending machine.\nGum: $.50, Peanuts:
$.55, Cheese Crackers: $.65, M&Ms: $1.00 \nEnter Item Selection: (G, P, C, or M) \
```

```
n"

.balign 4
userSelection: .asciz "\nYou selected %s. Is this correct (y/n)? \n"

.balign 4
userPayment: .asciz "\nEnter at least %d cents for selection.\nDimes(D),
Quarters(Q), and Dollar Bills(B): \n"

.balign 4
dispensed: .asciz "\n%s has been dispensed.\n"

.balign 4
enoughPayment: .asciz "\nAmount of money received is enough. \n"

.balign 4
noInventory: .asciz "\nOut of Stock!\n"

.balign 4
secretInventory: .asciz "\nGum - %d \nPeanuts - %d \nCheese Crackers - %d\n"

.balign 4
mnmInventory: .asciz "M&Ms - %d\n"

.balign 4
changeOutput: .asciz "\nChange of %d cents has been returned. \n"

.balign 4
itemGum: .asciz "gum"

.balign 4
itemPeanuts: .asciz "peanuts"

.balign 4
itemCrackers: .asciz "cheese crackers"

.balign 4
itemMNMs: .asciz "M&Ms"

.balign 4
ErrMsg: .asciz "Setup didn't work... Aborting...\n"

.balign 4
userInput: .asciz "%s"

.balign 4
strOutputNum: .asciz "%d \n"

.balign 4
strOutputArea: .asciz "\nArea: %d \n"

.balign 4
numInputPattern: .asciz "%d"

.balign 4
strInputPattern: .asciz "%[^\n]"

.balign 4
strInputError: .skip 100*4
```

```
.balign 4
numInput: .word 0


delayMs: .word 1000  @ Set delay for one second.
delay5Ms: .word 5000  @ Set delay for 5 second.

@ Define the values for the pins

pin2: .word 2

pin3: .word 3

pin4: .word 4

pin5: .word 5



.global printf
.global scanf

.extern wiringPiSetup

.extern delay

.extern digitalWrite

.extern pinMode

errorout:  @ Label only need if there is an error on board init.

@ End of code and end of file. Leave a blank line after this.
```